# A Framework for Quantitative Analysis of User-Generated Spatial Data

Tom Feltwell, Patrick Dickinson, Grzegorz Cielniak
School of Computer Science,
University of Lincoln, UK
email: tfeltwell,pdickinson,gcielniak@lincoln.ac.uk

**KEYWORDS**

Game Metrics, Spatial Data Analysis, First Person Shooter, Histogram Comparison

**ABSTRACT**

This paper proposes a new framework for automated analysis of game-play metrics for aiding game designers in finding out the critical aspects of the game caused by factors like design modifications, change in playing style, etc. The core of the algorithm measures similarity between spatial distribution of user generated in-game events and automatically ranks them in order of importance. The feasibility of the method is demonstrated on a data set collected from a modern, multiplayer First Person Shooter, together with application examples of its use. The proposed framework can be used to accompany traditional testing tools and make the game design process more efficient.

## INTRODUCTION

The constantly increasing complexity of modern video games poses new challenges for game designers. The design teams not only have to fulfil the growing demands with respect to quality, realism and detail of digital assets but also ensure that the game is fun, challenging and free from major loopholes. The design process is traditionally verified by different testing stages allowing for detection and identification of most critical shortcomings. Traditional testing tools usually involve questionnaire-based feedback or biofeedback provided by testers (Ambinder 2011) and analysis of so called game metrics  information generated by users during the actual game-play. The amount of information from the testing process necessary to be analysed by the designers is growing together with the complexity of games and can quickly lead to data overload, where there are too many factors to be considered and correlate with each other. Therefore there is a growing need for new automated methods for metrics analysis that are scalable and efficient.

This paper proposes a new framework for automated analysis of game-play metrics that will aid game designers in finding out the critical factors caused by factors like design modifications, change in playing style, etc. The core of the algorithm measures similarity between spatial distribution of critical game events and feeds that information back to the designer indicating critical differences in the generated data. The proposed framework can be used to accompany traditional tools and make the design process more efficient.

The paper is organised as follows: the next section describes relevant work in the area of quantitative analysis of user-generated data, then the details of the proposed framework are presented. The applicability of the method is then investigated in the experimental section followed by conclusions and discussion of future work.

## RELATED WORK

Statistical summaries of different in-game events generated by the player (also called game metrics) provide invaluable information to the game designers. For example, unbalanced weapon usage, high number of deaths or long level completion time can indicate serious design issues and can be quickly identified during the testing stage of the game development. Indirectly, this information can also provide insights into player's experience and can be used together with traditional methods for measuring player's experience that are based on subjective questionnaires or biofeedback (Davis et al. 2005, Tychsen 2008). The importance of statistical analysis of user generated data has been recently noted by the game industry and resulted in a number of existing systems for data collection and analysis (Kim et al. 2008, Wallner and Kriglstein 2012).

Many of the important in-game events relate to a specific location in the environment, such as defensive positions, attack routes and supply points. In such a case, not only overall frequency of these events is important but also their spatial distribution. Spatial histograms (or so called heatmaps) are a popular tool in video game industry for representing such data as they can be conveniently visualised and used for further inspection (Drachen 2011, Thompson 2007). Alternative approaches for spatial data visualisation include for exam-

ple clustering algorithms that group closely occurring events into meaningful nodes (Tychsen 2008).

The majority of the current methods for in-game spatial analysis rely on visual inspection by the game designer, who needs to take into account not only the shape of event distribution, but also *changes* in that distribution. With so many factors to be taken into account the process can be very tedious and time consuming. In this paper, we propose a step toward automated techniques for comparing and characterising changes in in-game event distribution. Our approach is inspired by techniques popular in other research areas like computer vision and pattern recognition. Example applications include image indexing (Swain and Ballard 1991), object tracking (Bradski and Kaehler 2008) and object recognition (Gevers and Smeulders 1997).


Figure 1: Screenshot from Red Orchestra: Ost Front 41-45. Image Copyright: Tripwire Interactive

## METHOD

This section presents details of the proposed system. First, the game scenario is presented that was chosen for illustrating concepts and techniques used in the system. However, the proposed techniques are not scenario, nor event specific and can be applied to any game type, single- or multi-player and any virtual environment where spatial data can be collected. The following descriptions present details of the four-stage framework for data processing (see Fig. 2). During stage one (S1), spatially distributed data is collected and suitably formatted from a virtual environment/game. Stage two (S2) sees the generation of spatial distribution of various events in a form of a histogram. Different histograms are then compared in stage three (S3) and finally the results of these comparisons are ranked and characterised in stage four (S4).

### Game Scenario

The proposed game scenario is Red Orchestra: Ost Front 41-45 (see Fig. 1). Developed in 2006 by Tripwire Interactive, the player takes charge of various roles within either a Russian or German military unit on the Eastern Front during World War 2 (Tripwire Interactive 2006). The game is a purely multiplayer First Person Shooter (FPS), with heavy emphasis on team work. Within each team, a set of player "classes" are available, which each player must chose from before commencing play. These classes are designed to encourage the team playing aspect, as they each have unique weapons and abilities. Furthermore, some classes such as Assault Trooper are restricted to maintain balance of the multiplayer combat, and to represent realistic military squad hierarchy.

Realism is one of the key features of the game, with many simulation aspects implemented, such as bullet flight time, wound ballistics and bullet drop. Due to the complex nature of the game system, it was decided that the game would provide a good source of data for characterisation and analysis. Based on the Unreal 2.5 Engine, support and documentation was readily available via Epic Games (2004). Further more, an open source Software Development Kit was available for Red Orchestra, allowing full access to source code and level editor, which is well support by both the community and Tripwire Interactive.

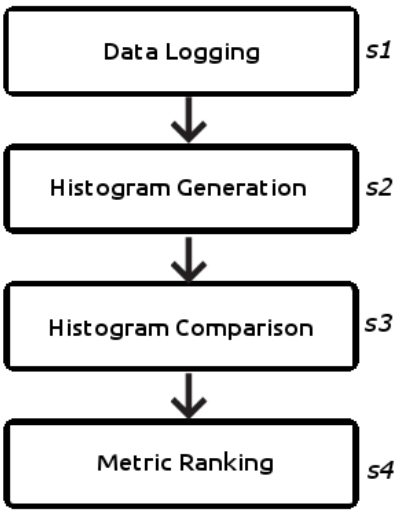


Figure 2: Stages of the Proposed Framework

### Data Logging (*S1*)

The initial stage of the framework is the collection of relevant data for analysis. The collected data must have a spatial element associated within it. Most virtual environments use a coordinate system, where the spatial element of the data would consist of X,Y and Z coordinates. Some examples of spatially distributed events include the change of direction of the car wheels in a racing game or the combat engagement between two units in a real time strategy. In Red Orchestra, the players typically perform five basic actions: shooting, moving, dying, taking damage and reloading. Other relevant information related to these key events include crouching, sprinting and using a specific weapon. These events were therefore chosen for logging and further analysis.

Logging of so called *user initiated events* is commonly used in the industry and is known as *instrumentation* (Kim et al. 2008). The data logging system created for the proposed system was written in UnrealScript, the native language of Unreal Engine 2.5. In our implementation the game server stores and records all logs, with no log information transferred across the network. Per match, each player has their own log file, with a unique file name. There is an overall match log, which stores events such as changing team or capturing objectives - these are however not spatially distributed events and are recorded only as additional information. To record Move events, the player's position is polled in regular intervals (1 s. in the proposed implementation). All other events are recorded immediately as they happen. Each event contains a standard set of information: position, player name, time, event type and player stance. Depending which event is being recorded, an extra event specific set of information is appended onto the log entry. The basic event log is formatted as follows: *eventType* | *eventTime* | *playerName* | *X,Y,Z coords* | *crouched* | *prone* | *sprinting* | *limping* | *stamina*. An example of an actual event log can be seen in Fig. (3.

```
1|17:24:23|Stinkfoot|1829.50,-4232.90,283.30|1|0|0|False|13.97
1|17:24:24|Stinkfoot|1829.78,-4262.36,287.18|1|0|0|False|14.92
1|17:24:25|Stinkfoot|1859.57,-4226.42,284.05|1|0|0|False|15.79
0|17:24:25|Stinkfoot|1880.33,-4203.43,282.73|1|0|0|RO-LyesKrovy.STG44Weapon|1|0,654
1|17:24:26|Stinkfoot|1880.33,-4203.43,282.73|1|0|0|False|16.60
4|17:24:26|Stinkfoot|1880.33,-4203.43,282.73|1|0|0|el_mysterio|1883.69,-4188.46,306
0|17:24:26|Stinkfoot|1880.33,-4203.43,282.73|1|0|0|RO-LyesKrovy.STG44Weapon|1|0,118
0|17:24:26|Stinkfoot|1880.33,-4203.43,282.73|1|0|0|RO-LyesKrovy.STG44Weapon|1|0,42,
0|17:24:26|Stinkfoot|1880.33,-4203.43,282.73|1|0|0|RO-LyesKrovy.STG44Weapon|1|0,132
0|17:24:26|Stinkfoot|1880.09,-4204.52,282.73|1|0|0|RO-LyesKrovy.STG44Weapon|1|0,44,
1|17:24:27|Stinkfoot|1879.34,-4205.52,282.73|1|0|0|False|17.56
1|17:24:28|Stinkfoot|1859.37,-4256.04,287.82|1|0|0|False|18.39
1|17:24:29|Stinkfoot|1868.35,-4373.83,297.36|1|0|0|False|18.04
```

Figure 3: Example Player Event Log collected during a Data Gathering Session

Python scripts were written to parse the events contained within the log files and to store them in a SQL database, which encapsulates the events, game matches and other information. SQL format was chosen due to its popularity, simple syntax of queries and built-in optimisation for handling large numbers of records. The file names of the logs contain unique information such as time the player joined, map being played and player name. These are used to populate the SQL database initially.

### Histogram Generation (*S2*)

The next step of the procedure is generation of spatial histograms from the collected data. A spatial histogram is a discrete representation of the environment, 2d in the presented case, but can easily be extended to 3d, and consisting of $n \times m$ so called bins. Each bin corresponds to a rectangular region of the environment and the value of each bin stores the total count (i.e. frequency) of the specific event that occurred in that region (see Fig. 4). In result, the spatial histogram not only provides information about the frequencies of events, but also their *spatial distribution*.

The resulting histogram can be represented as a "heatmap" - a histogram using "hot" and "cold" colours mapped to the high and low (respectively) frequency values. This gives the impression that areas with higher event counts are "hotter" than low event counts, which are "colder" (see Fig. 4c and 4d).

Two critical parameters affecting the shape of the resulting histograms are the size of the region corresponding to each bin, and the total number of events considered. Assuming that the size of the environment is fixed, larger bin size results in fewer bins or in other words, in lower histogram resolution (see Fig. 4d). On the other hand, larger bins require less events to populate the bins (i.e., result in sufficient count of events).

There are existing methods that calculate the optimal bin size depending on underlying data distribution (Shimazaki and Shinomoto 2007). However, they are usually based on some strong assumptions (e.g. Normal distribution) which do not hold in all cases. Therefore in the proposed system the number of bins is an experimentally tuned parameter.

### Histogram Comparison (*S3*)

Spatial histograms of different events or generated under different conditions can then be compared to provide a single metric value that indicates their similarity/dissimilarity. Examples of different conditions are data from different time periods, (i.e. first five games compared to last five games), selection of matches with particular player numbers, or with differing sample populations, such as different teams of players. Before the comparison, each histogram has to be normalised by dividing the value of each bin by the total sum of all bins. The resulting histogram becomes a discrete approximation of probability distribution for the event. It is im-

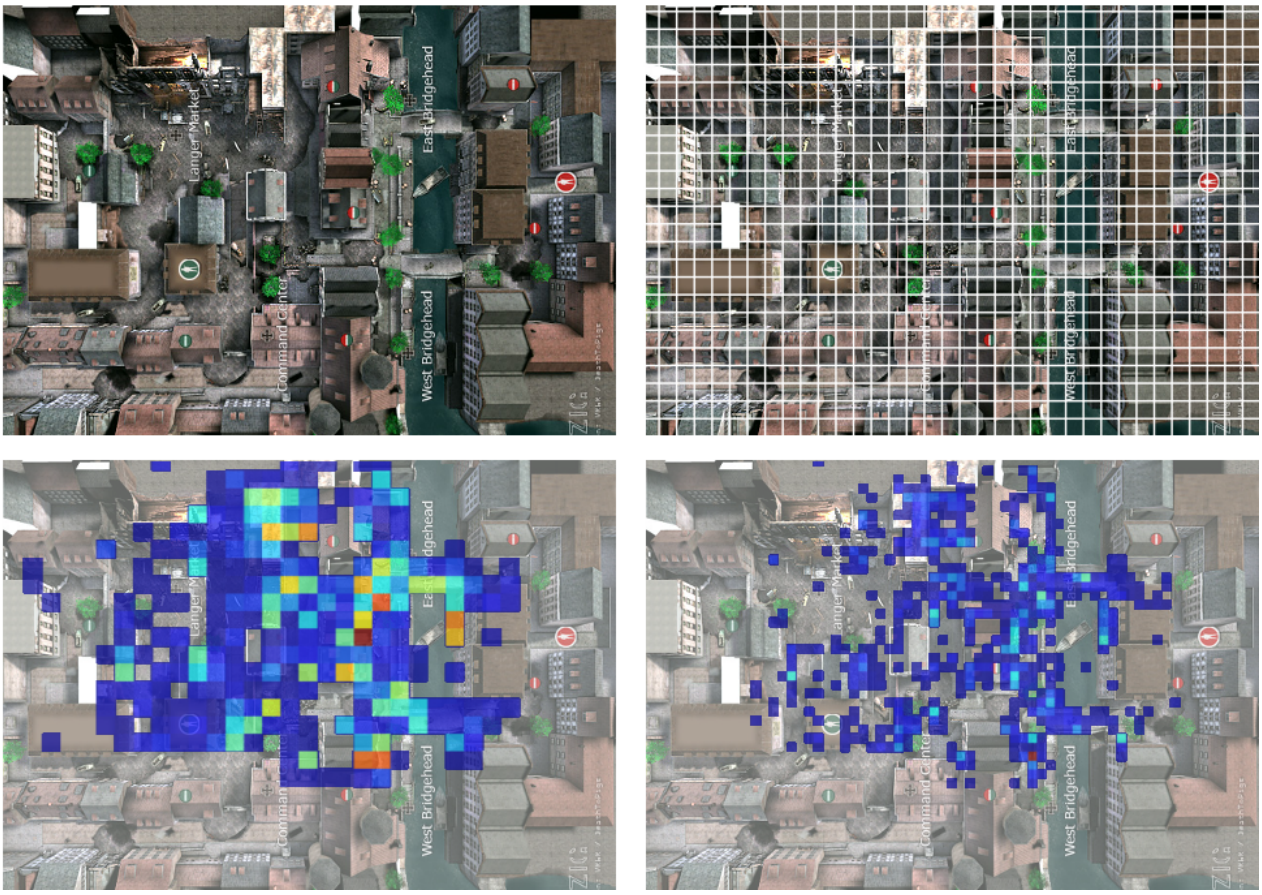


Figure 4: a) Danzig Map b) Danzig Map with White Grid Lines Depicting Histogram Bins c) The Overlaid Histogram of Shoot events (32 Bins) d) The Same Histogram with Higher Resolution (64 Bins)

portant to note that most comparison methods require histograms of the same resolution.

There are several methods that can be used for comparison of histograms (Abramowitz and Stegun 1965). Here, two popular metrics are proposed. The $L_1$ metric measures the amount of difference between the histograms by summing together all absolute differences between the histograms bins:

$$L_1 = \sum_{i=1}^{B} |P(b_i) - Q(b_i)|, \tag{1}$$

where $P(b_i)$ and $Q(b_i)$ are the normalised values stored in the i-th bin of the P and Q histogram, respectively, and $B = n \times m$ is the total number of bins for each histogram. The resultant metric can assume values from [0-2] range, with 0 value corresponding to two identical histograms.

A metric based on Bhattacharyya distance is used to measure the amount of overlap between two histograms (Dubuisson 2010, Thacker et al. 1997). The modified formula that expresses that value in terms of similarity for [0-1] range is as follows:

$$L_{Bh} = 1 - \sum_{i=1}^{B} \sqrt{P(b_i)Q(b_i)}. \tag{2}$$

### Metric Ranking (*S4*)

The proposed comparison metrics summarise differences of a pair of histograms in a single number. Metric values can then be directly compared and provide instant information about the amount of change for each condition/event type. Higher metric values might indicate significant changes in critical variables while lower metric values might help identifying similar patterns occurring in the data. This information can direct the designer in further analysis of the results, for example by visual inspection. This way the designer avoids looking at all various possible combinations of histograms ("data overload") and can concentrate on the critical variables only.

## EXPERIMENTS

The following experiments were conducted to demonstrate the feasibility of the proposed framework. This section presents details about data gathering, data sets collected and results obtained from applying the presented framework to the selected game scenario.

### Data Collection

Figure 5: One of the Data Gathering Sessions

The data collection was scheduled as regular sessions in a designated area of the computer lab at the Lincoln School of Computer Science (see Fig. 5). The sessions were advertised amongst the School's staff and students. Unfortunately, due to the restrictive University network policies, it was not possible to collect data through the Internet and address a wider audience.

Each session consisted of several multiplayer matches of Red Orchestra: Ost Front 41-45, with players joining one of the two available teams, such that matches were approximately balanced. Each session was held for approximately two hours, with three maps (Danzig, Basovka and Lyes Krovy) played in a randomised order every time. Approximately 50 sessions were run over the duration of one year, with most sessions taking place during term time, due to staff and student availability.

From all available data we have chosen a representative subset consisting of 30 matches on a single map, Danzig. The number of players per match varied between 5 and 20 (see Fig. 6). The events considered for further analysis include the 5 aforementioned types including Shoot, Move, Death, Damage and Reload. The average of each event per player can be seen in Fig. (7). The collected data was split roughly in half into two sets, $D_1$ and $D_2$ for further analysis.

$D_1$ contained 88,048 data points, and $D_2$ contained 98,701 data points. The breakdown of these data sets into their respective events can be seen in Table (1). The difference between event counts for the two data

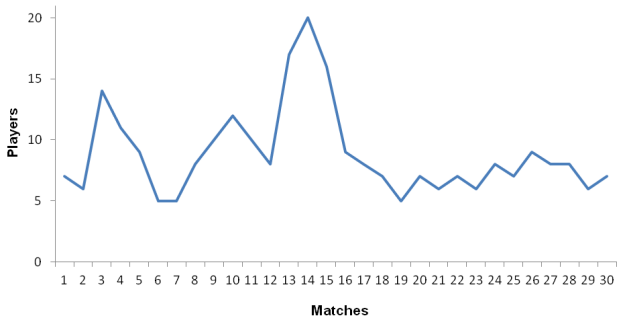


Figure 6: Player Count Per Match over the Entire Data Set

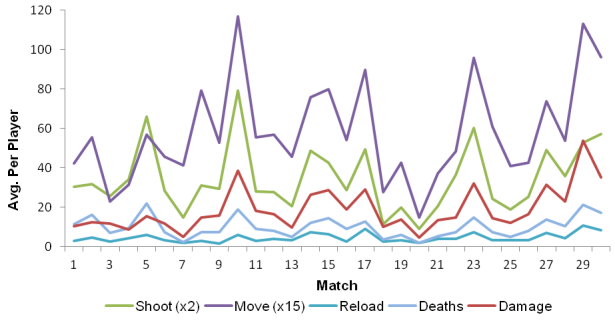


Figure 7: Average Event Count Per Player Per Match over the Entire Data Set

sets can be attributed to multiple factors, for example strong correlation can be seen between the number of players (see Fig. 6) and event counts (see Fig. 7). The Reload and Shoot event are correlated to the type of weaponry being used by players, as some weapons require reloading more often and some are able to fire shots more quickly than others.

Table 1: Total Number of Events in each Data Set

| | Shoot | Move | Reload | Death | Damage |
|---|---|---|---|---|---|
| $D_1$ | 5,968 | 79,111 | 391 | 844 | 1,734 |
| $D_2$ | 6,661 | 88,435 | 513 | 955 | 2,137 |

### Results

With $D_1$ and $D_2$ data sets isolated, comparisons were generated for each of the five events including both proposed metrics. Fig. (8) presents $L_1$ and $L_{Bh}$ metric values for each event type with respect to varying histogram resolution. The bin size parameter varied from 2 to 64 bins along the X axis; the corresponding number of bins along the Y axis has been chosen such that the resulting regions were square (e.g. for Danzig $64 \times 44$ bins) It can be seen that for small bin size the difference between different events is difficult if not impossible to

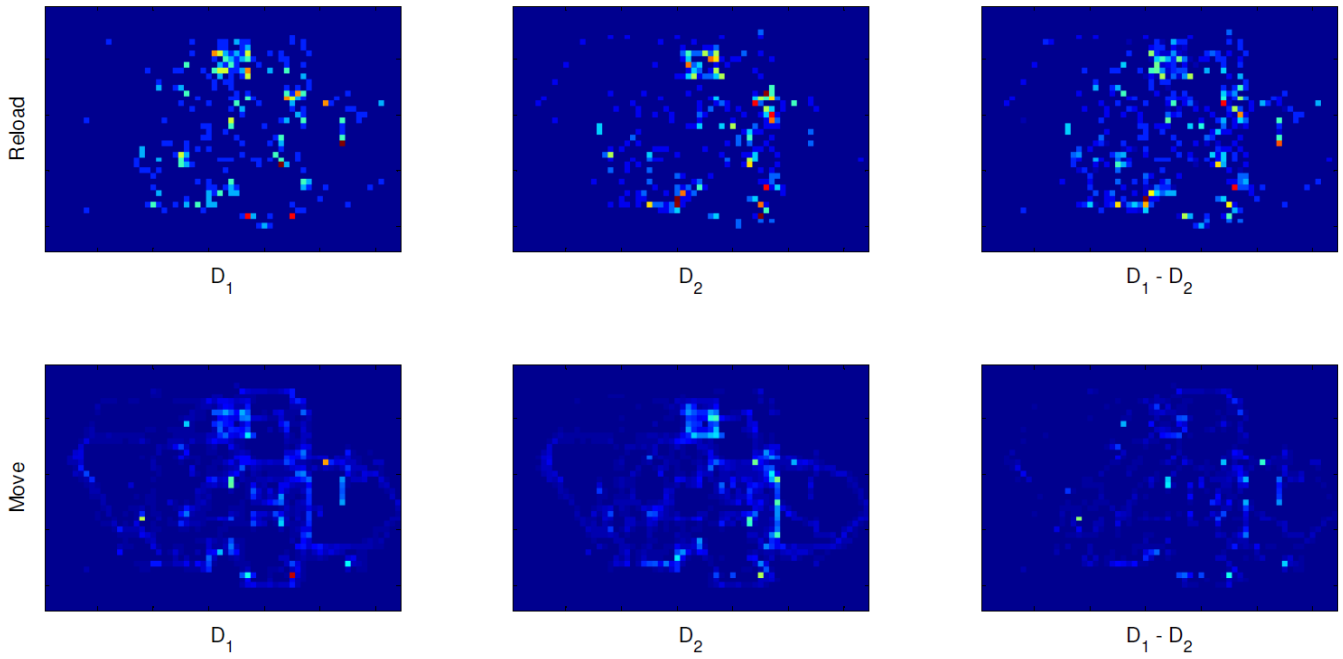


Figure 9: $D_1$, $D_2$ Histograms for Reload and Move Events, and their Differences ($64 \times 44$)

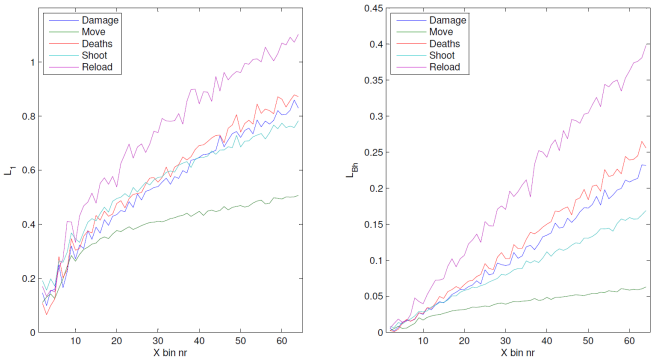


Figure 8: $L_1$ and $L_{Bh}$ Comparisons over Varying Histogram Resolution

identify, due to low resolution of spatial histograms. As the number of bins increases, metric values for the different events take on their own separate characteristics. Over the remainder of the range, they maintain similar relationship to one another. With 10 bins and or less, Bhattacharyya metric presents itself as more stable, when compared to $L_1$. It is interesting to note in both comparison metrics, the relationship between events is broadly similar from 10 bins upward.

The results indicate that metrics for the Reload event have the highest value, which is due to large differences in the two distributions. The movement event has the lowest metric values, which might indicate similar patterns in distribution for this event. Fig. (9) provides visual guidance in the form of plotted histograms for the Reload and Move events, confirming the presented metric results. It can be seen that indeed changes in the Reload event are more visible than for the Move event. It can be seen for Reload that the main cluster at the top of the map in $D_1$ has shrunk in size in $D_2$, with another cluster appearing on the right. These two clusters match up with defensible areas of the Danzig map, indicating players defending/attacking these in different ways between the two data sets. This is validated by the comparison, which shows difference in both locations.

For Move events, "pathways" can be seen in $D_1$ and $D_2$ histograms, which indicate the walkable areas of the levels. Some pathways are used more often than others, and are represented on the histograms in "hotter" colours. The bright clusters seen on both $D_1$ and $D_2$ are areas where players spend more time. These points correspond to vantage points, spawn areas and objectives within the game world. It is interesting to note that the comparison histogram for the Move event shows overall a large number of small differences. These can be explained by gameplay conditions and player behaviour. A small number of high value differences are also present, which represent particular vantage points visited frequently by players.

The presented framework can also be used to observe temporal changes of various events. To demonstrate that functionality, the entire data set ($D_1 + D_2$) was divided into 8 distinct sets, and comparisons were run for each possible pair of the resulting subsets. Table (2) presents a rectangular similarity/dissimilarity matrix for each of the 8 data sets, using $L_1$ metric for the

Table 2: $D_1$ and $D_2$ Divided into 8 Sections, Compared with each Section using $L_1$ Metric, for the Move Event

| | $D_1^1$ | $D_1^2$ | $D_1^3$ | $D_1^4$ | $D_2^1$ | $D_2^2$ | $D_2^3$ | $D_2^4$ |
|---|---|---|---|---|---|---|---|---|
| $D_1^1$ | 0.00 | 0.91 | 0.86 | 0.86 | 0.92 | 0.92 | 0.79 | 0.95 |
| $D_1^2$ | | 0.00 | 0.77 | 0.84 | 0.85 | 0.82 | 0.80 | 0.86 |
| $D_1^3$ | | | 0.00 | 0.79 | 0.86 | 0.76 | 0.76 | 0.83 |
| $D_1^4$ | | | | 0.00 | 0.81 | 0.90 | 0.79 | 0.93 |
| $D_2^1$ | | | | | 0.00 | 0.73 | 0.71 | 0.80 |
| $D_2^2$ | | | | | | 0.00 | 0.70 | 0.70 |
| $D_2^3$ | | | | | | | 0.00 | 0.75 |
| $D_2^4$ | | | | | | | | 0.00 |

Move event. It can be noted the dissimilarities are lower for adjacent subsets from the same set (e.g., $D_1^2 - D_1^3$, $D_2^2 - D_2^3$) and in general are higher for subsets further apart (e.g. $D_1^1 - D_2^4$). Such a matrix can be used to identify similar or reoccurring event distributions for different time intervals.

Perhaps the most interesting comparison is for adjacent regions. Fig. (10) plots the comparison values for adjacent regions for each event type. It is interesting to note the relationships and changes between each metric over time, with Death and Damage events showing a small degree of an inverse relationship to the other metrics.

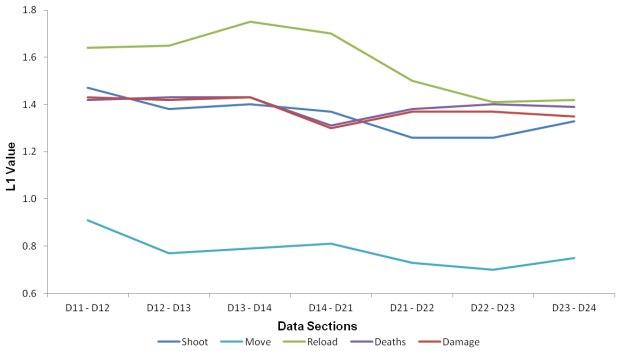


Figure 10: Comparison Metrics for Adjacent Data Sections for Different Event Types

## CONCLUSIONS

This paper introduced a framework for quantitative analysis of user-generated spatial data. The presented experiments demonstrate the feasibility of the proposed method for quantifying differences between different data sets and provide example scenarios for its use. However, the presented work is still in its initial stage and further investigation is needed to address its full potential. With possibility of quantifying changes in two different data sets, it should be possible for example, to detect and identify design changes introduced by the game designer or analyse typical distributions of events for different player teams (e.g. beginners vs. advanced). This step would close the loop in the system by providing an instant feedback mechanism to the designer regarding the impact of their introduced changes on different gameplay factors. The proposed metrics provide global summaries of changes in distributions. Further work will seek to automatically identify specific regions where the most changes occur. The influence of other factors, like for example the number of players per match or a total number of events on the resulting metrics should also be investigated.

## REFERENCES


M. Abramowitz and I. Stegun. *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*. Dover Publications, 1965.

M. Ambinder. Biofeedback in gameplay : How valve measures physiology to enhance gaming experience. *Game Developers Conference*, 2011.

G. Bradski and A. Kaehler. *Learning OpenCV: Computer Vision with the OpenCV Library*. O'Reilly, Cambridge, MA, 2008.

J. P. Davis, K. Steury, and R. Pagulayan. A survey method for assessing perceptions of a game: The consumer playtest in game design. *Game Studies*, 5:1–14, 2005.

A. Drachen. Evaluating motion: spatial user behaviour in virtual environments. *International Journal of Arts and Technology*, 4:294–314, 2011.

S. Dubuisson. The computation of the bhattacharyya distance between histograms without histograms. *Image Processing Theory Tools and Applications IPTA 2010 2nd International Conference on (2010)*, 2010.

Epic Games. Unreal Engine 2.5. http://www. unrealengine.com, 2004.

T. Gevers and A. W. M. Smeulders. Color based object recognition. *Pattern Recognition*, 32:453–464, 1997.

J. H. Kim, D. V. Gunn, E. Schuh, B. C. Phillips, R. J. Pagulayan, and D. Wixon. Tracking real-time user experience (true): A comprehensive instrumentation solution for complex systems. *Proceeding of the twentysixth annual CHI conference CHI 08*, pages 443–451, 2008.

H. Shimazaki and S. Shinomoto. A method for selecting the bin size of a time histogram. *Neural Computation*, 19:1503–1527, 2007.

M. J. Swain and D. H. Ballard. Color indexing. *International Journal of Computer Vision*, 7:11–32, 1991.

N. A. Thacker, F. J. Aherne, and P. I. Rockett. The bhattacharyya metric as an absolute similarity measure for frequency coded data. *Kybernetika*, 34:363–368, 1997.

C. Thompson. Halo 3 : How microsoft labs invented a new science of play. *Wired*, 15:1–7, 2007.

Tripwire Interactive. Red Orchestra: Ost Front 41-45. http://www.redorchestragame.com, 2006.

A. Tychsen. Crafting user experience via game metrics analysis. *NORDICHI 2008*, pages 1–5, 2008.

G. Wallner and S. Kriglstein. A spatiotemporal visualization approach for the analysis of gameplay data. *Proceedings of the 2012 ACM annual conference on Human Factors in Computing Systems*, pages 1115–1124, 2012.

## AUTHOR BIOGRAPHY

**TOM FELTWELL** completed his B.Sc in Games Computing at the University of Lincoln in 2010. Upon his graduation he commenced the Masters by Research program at the same institute. He now works full time within the School of Computer Science as a Technician and is completing his degree part-time. His research areas are games related, particularly in the areas of simulation, realism and gameplay analysis. `tfeltwell@lincoln.ac.uk`

**PATRICK DICKINSON** has been a Senior Lecturer at the University of Lincoln since 2008. He has a Ph.D from Lincoln for his work in computer vision, and has previously studied at the Universities of Oxford and Southampton. His current research interests include the application of computer vision to the surveillance of wildlife, and games AI. He has previously worked at Rebellion Developments, and Awesome Developments, and worked on a number of published game titles. `pdickinson@lincoln.ac.uk`

**GRZEGORZ CIELNIAK** is a Senior Lecturer at the School of Computer Science, University of Lincoln, UK. He obtained his Ph.D. in Computer Science from Örebro University, Sweden in 2007 and his M.Sc. in Robotics from Wrocław University of Technology, Poland in 2000. His research interests include physics simulation and metric-based assessment of game play, mobile robotics, machine perception, behaviour analysis and monitoring of humans and other species. `gcielniak@lincoln.ac.uk`